UDK 006.72 **DOI** 10.15622/ia.2021.3.5

S. ALTAF, S. IQBAL, M. SOOMRO

EFFICIENT ALGORITHM FOR NATURAL LANGUAGE CLASSIFICATION TO DETECT DUPLICATE UNSUPERVISED FEATURES

Altaf S., Iqbal S., Soomro M. **Efficient Algorithm for Natural Language Classification to Detect Duplicate Unsupervised Features.**

Abstract. This paper focuses on capturing the meaning of Natural Language Understanding (NLU) text features to detect the duplicate unsupervised features. The NLU features are compared with lexical approaches to prove the suitable classification technique. The transfer-learning approach is utilized to train the extraction of features on the Semantic Textual Similarity (STS) task. All features are evaluated with two types of datasets that belong to Bosch bug and Wikipedia article reports. This study aims to structure the recent research efforts by comparing NLU concepts for featuring semantics of text and applying it to IR.

The main contribution of this paper is a comparative study of semantic similarity measurements. The experimental results demonstrate the Term Frequency–Inverse Document Frequency (TF-IDF) feature results on both datasets with reasonable vocabulary size. It indicates that the Bidirectional Long Short-Term Memory (BiLSTM) can learn the structure of a sentence to improve the classification.

Keywords: Clustering, Information Retrieval, TF-IDF Feature, Par2Vec, Natural Language Texts, Lexical Approaches

1. Introduction. Humans can exchange all kinds of information through language. For instance, language is used to talk about activities, discuss abstract concepts, and even helps to determine the sentimental state of other human beings. Consequently, language is used frequently in all kinds of communication, like e-mails, reports, conversations, and scientific papers. This usage generates large language datasets, which contain a lot of information. Approximately 80% of all relevant corporate data is text-heavy unstructured data [1]. Unfortunately acquired insights out of the text are a major problem for big data approaches. Currently, the main part of that information cannot be used for further automatic analysis. The lack of such methods makes it difficult to find relevant information in large text datasets.

Without being able to find relevant information already existing information is likely reproduced. The reproduction of information causes regularly unnecessary work. One example of this is the reimplementation of an algorithm. In software development, it is well known that this duplication creates additional problems for maintaining the implementations and even causes inconsistencies based on slight differences in the several implementations. The produced heterogeneous data is a problem in other areas as well and makes the search itself more difficult. Other consequences of information shortage are even more devastating. Managers could make wrong business decisions, lawyers would not be able to defend their clients, engineers could design buggy products

and the research of scientists would be more difficult without accessing the results of colleagues. The information has become by far the most valuable resource for almost everyone. Therefore, the required access through Information Retrieval (IR) systems to manage large amounts of data is one of today's key challenges. IR recognizes the implicit patterns contained in collections of unstructured data to find data that satisfy information needs [2]. This defines a variety of IR algorithms for a broad application field. Depending on the task particular datasets, patterns, and formulations of information needs are considered.

The main objective of IR is making information accessible through document search based on an information requirement. It organizes large amounts of data and structures the data according to implicit patterns contained in the data and the information required. IR tasks consist of two distinct sub-goals: understanding the input data and analyzing it concerning this understanding by classification or clustering methods. Understanding a text according to Wittgenstein means creating a picture based on this text and a priori knowledge [3]. Transferring this concept into the domain of machine learning the creation of a picture is the generation of representation within a vector space model based on several features of the text. Present available systems depend on several domain-specific features and so recognize certain features [4]. The research moves towards a general understanding of texts to create a task-independent representation of the meaning similar to the pictures humans create in their minds.

After the NLU algorithms created a representation for all texts, it is possible to analyze them regarding specific patterns. This analysis step provides a subset of documents, which comply with information needs: the retrieved documents [3]. Therefore, the analysis focuses on grouping and potentially ranking documents according to this need. Information needs can either be formulated directly by transforming them into an NLU feature and compared them to the document representations or implicitly by classifying all documents [4]. This text classification determines information needs based on different classes and requires labeled data for each class. In contrast to grouping all documents according to labels, the direct comparison of NLU features does not depend on labeled data. Instead, the most similar documents compared to information needs are retrieved by clustering them. Clustering gathers similar documents and information needs with a data-driven approach. It does not need a priori knowledge about the partitioning of the document collection [5]. However, the main goal of retrieving documents, which satisfy a specific information need, is shared by both supervised and unsupervised models. The retrieval task determines whether classification or clustering is applied. The

following (Fig. 1) depicts the fundamental IR process with the described NLU and analysis steps is given by.

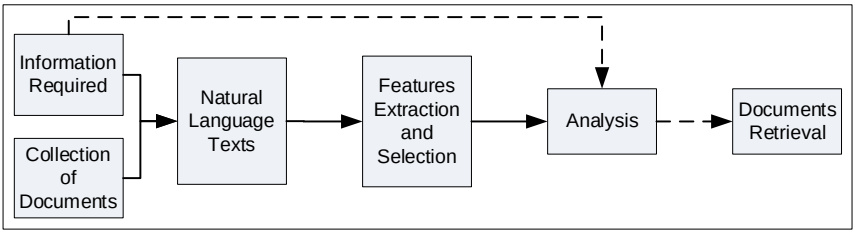


Fig. 1. Schematic diagram of the transformation of text into the feature extraction and selection with NLU

This work aims to structure the recent research efforts by comparing NLU concepts for featuring semantics of text and applying them to IR [6]. Following this goal, the main contribution of this paper is a comparative study of semantic similarity measurements. The focus of this study is to identify useful patterns and heuristics for specific IR tasks with varying datasets.

The rest of the paper is organized in six sections as follows. Section (2) provides an overview of the state-of-art related work that leads to motivation and problem identification in section (3). Section (4) presents the Natural Language Texts Features and their methodologies leading towards the Lexical Approaches in Section (5). Section (6) presented on Transfer-Learning approach and experimental results based on different discussed algorithms in section (7) and the conclusion of this paper are presented in the following section (8).

2. Literature Review. The field of Information Retrieval is a cornerstone of the current information age and has a significant value for corporations and individuals. This value becomes apparent by looking at the success of web search engines, which represent just one application of the whole IR field. The diverse field of IR systems has led to many publications around this topic. The described state-of-the-art in the following sections focuses on the creation of a vector space model to represent the meaning of texts and the related classification and clustering analysis tasks.

2.1. Natural Language Understanding. As described the first step towards finding relevant information understands the document collection and the information needs. The field of NLU addresses this problem for general NLP tasks by extracting features, which represent the meaning [6]. Those features can be extracted either by rule-based systems or with data driven statistical approaches. Similar to other fields in Artificial Intelligence (AI) recent algorithms of NLU focus on statistical deep learning models, which provide

numerical features instead of simple nominal features provided by most rule-based approaches [7].

2.2. Rule-Based Approaches. First NLU approaches were published between 1960 and 1975. The examples of those early systems are solving algebra text problems from schoolbooks [8] or answering the question about the position of building blocks in a model world [5] to prove the understanding of text input. Those early systems use rule-based methods to understand language commands and perform actions according to understanding. For example, the block model of the SHRDLU system [9] moves blocks and answers questions based on English keywords. The rules used in NLP systems are described with context-free grammars and checked by parsers. For example, SHRDLU uses systemic functional grammar to process language commands [10]. The idea of rule-based systems is supported by the linguistic concept of generativist. This concept was introduced by Noam Chomsky and claims that general laws and principles govern all NL, therefore NL can be described with a generative grammar [10].

Rule-based systems focus on a specific domain, like a restricted world of colored blocks. In this domain, the possible inputs and outputs can be processed with few rules or simple grammar and the system can fully understand the reduced language input. Improvements in the following years showed that those systems are very difficult to scale to a wider domain or adapt to another domain [11]. Rule-based systems, like SHRDLU [9], require knowledge about the different utterances and how they can be connected. Therefore, such systems grow exponentially with an increasing number of utterances. Nevertheless, rule-based approaches are still considered in NLP applications with specific domains, especially data preparation relies on the simplistic approaches provided by rule-based systems.

2.3. Statistical Approaches. The increase in computational power and available language data was responsible for the shift from rule-based methods to data-driven statistical methods in the early 1990s. The statistical revolution also marks the shift from generative grammars to more structural linguistics-based approaches. Structural linguistics focus on probabilities assembled over a large utterance corpus [12]. Instead of trying to find complex rules, which describe the language, statistical methods analyze the occurrence of words and phrases and try to find patterns to understand the text. An example of this analysis task is language models. Those models try to predict a word with the help of given context words. That so-called word embedding is proven to be useful for many NLP tasks. The language models are used as inputs for supervised NLU tasks, like Question Answering (QA) [54], sentiment analysis [13], or Semantic Textual Similarity (STS) [14]. Those systems use Convolutional Neural Networks (CNNs) or Recurrent Neural Network Training (RNNs) with large training datasets and handcrafted additional features like Part-of-Speech (POS) tags.

Other NLP tasks like machine translation, text summarization use similar encoder-decoder architectures to convert the text. Those tasks also require a semantical understanding of the input text; therefore, the encoder is also trained to produce an abstract representation of the meaning [13]. Statistic methods can process the complete corpus of a language. They are not restricted to specific domains with reduced vocabulary [15]. However, statistical methods focus on specific subtasks of NLP, like language models instead of understanding the whole language.

Some research related to the transformer models has been presented [9] to show the overall representation of network layers and they are based on Transformers Architecture. Most of QA models in the field of NLP are encoder-decoder based architectures. QA has applications in a vast array of tasks including information retrieval, entity extraction, chatbots, and dialogue systems to name but a few. While QA can be done in various ways, perhaps the most common flavour of QA is selecting the answer from a given context. In other words, the system will pick a span of text from the context that correctly answers the question. If a correct answer cannot be found from the context, the system will merely return an empty string. BERT [22], XLNET [23], CoQA [24] and QuAC [25] are all commonly used models for QA. Sur et al. [26], proposed a model on google based BERT model by adopting the BERT linear output layer including decoding framework for overall analysis of IR.

2.4. Thesauri-Based Approaches. Thesauri-based approaches formulate specific rules based on the a priori relations. The features are nominal, and it is only possible to analyze words within a thesaurus with predefined relations. Naturally, words are nominal [16] and distributional semantics introduces a notion of similarity based on the differences of the word co-occurrences. This is used by distributional models to generate numerical features. Co-occurrences are not able to distinguish word relations and therefore use vague relations. The thesauri-based approaches mostly rely on manual feature engineering. Therefore, it does not use feature learning at all. Subsequently, no additional training data is required. The landscape of distributional features is more divided. Traditional models use unsupervised methods, like Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) to reduce the dimensionality. Word embeddings use language models and thereby are also unsupervised. Finally, typical deep learning approaches, like CNNs or RNNs integrate the feature learning into the classification task [14]. Essentially each layer of those networks represents features. To learn those features labeled data is required. The features are closely related to the classification task they are trained on and do not provide any theoretical insight. Thesauri-based approaches are not considered, because differentiating between special word relations is not required and distributional models evolved as quasi-standard for the vast majority of NLP tasks [4]. Also,

creating a suitable thesaurus based on the specific domain data is time-consuming, costs expensive, and would require an extensive domain and linguistic knowledge [3]. As a consequence, the subsequently presented methods are restricted to distributional semantics.

2.5. Classification. In IR, every search can be seen as a classification problem. The classification separates all documents into a group, which matches an information need, and another group of documents, which does not match the information, need [16]. Consequently, it groups the collection constantly into multiple groups. Each group satisfy one specific information need. This makes classification suitable for common search terms and split a collection into several known topics. This is also the reason why classification in IR is referred to topic classification. Typical IR classification tasks are spam filtering or finding news stories about a certain topic. However, tasks like sentimental analysis can be considered as retrieval tasks as well. Most algorithms use word-level features to classify text. Term frequency extensions are used to create feature invariant to text length and take the different information entropy of words into account. Chali et al. [16] introduced tree structured RNNs according to dependency graphs to group the sentiment of movie reviews. Aside from word-level representation CNNs, which works for many tasks well on raw signals, are used to classify the collection on character level [10]. This approach could recognize the meaning of affixes.

2.6. Clustering. In IR, clustering is based on the so-called cluster Theoretical hypopaper testing. Aswani et al. [17] describe the hypopaper as follows: "Documents in the same cluster behave similarly concerning relevance to information needs"[17]. Accordingly, if one document of the cluster satisfies an information need, all other documents of the cluster most likely satisfy it as well. This hypopaper can be exploited for many use-cases in IR. One example is applying hierarchical clustering to stepwise refine the information need. At each level, it is possible to select the cluster most suitable for the intended information need. This scatter-gather search avoids the ambiguity of search queries [18]. Another example of clustering is speeding up the search process by dividing the collection into clusters and searching just the nearest clusters to the information need. The used clustering algorithms are determined by the specific, demanded characteristics of the clustering task, therefore the clustering approaches are highly tasked specific. Commonly k-means models are used for topic clustering and scatter-gather searches often use single-linkage approaches [18]. However, most applications require specialized clustering algorithms to take unknown cardinalities, partial or fuzzy cluster, and prototype representations of clusters into account.

3. Problems Identification. Section 2 reveals a few limitations of the NLU feature and IR. Based on the following depicted limitations the contributions of this work are going to be explained here.

3.1. Natural Language Texts Feature. The current research on the semantic feature space for IR tasks has two important shortcomings, which will have been addressed by this paper. Firstly, the end-to-end approaches applied by many supervised NLP tasks like STS, sentiment analysis, or QA rely on training data and do not provide an explicit sentence or even text representations. Therefore, unsupervised tasks have to fall back on simpler features [19]. The recently proposed general-purpose representations based on transfer learning tasks could solve this problem, but as of today, those features were not used for IR tasks. Secondly, feature taking the word order into account are trained and used mostly on grammatically correct sentences. In IR, many documents do not contain just continuous text. Bullet points, keywords, and other interruptions of text with a distinct grammatical structure are common. Consequentially the current research does not report results for those noise data.

3.2. Datasets Analysis. Current text classification approaches use the large dataset with a structure aligned to English grammar rules. Therefore, those models lack evaluation capabilities for less strictly structured text with smaller training datasets. Due to the predominantly unsupervised setting, most IR approaches still rely on the simple feature, which is obtained in an unsupervised fashion [20]. Those term-frequency or word embedding models fail to represent the combined meaning of words in a sentence.

4. Natural Language Texts Feature. Natural Language Understanding is a large research field with an even wider range of tasks, each of them with a unique set of requirements and varying definitions of understanding. Translating a text requires another form of understanding than identifying the overall topic of a text or answering questions. Based on those differences each task also requires specific NLU features. This is the reason why many features with divided characteristics exist. To select, evaluate, and even understand those features is useful to categorize them. Distributional models require instead of a priori defined relations between certain words a large text corpus to obtain the implicit relations. The features are nominal, and it is only possible to analyze words within a thesaurus with pre-defined relations. Naturally, words are nominal. Distributional semantics introduces a notion of similarity based on the differences of the word co-occurrences. This is used by distributional models to generate numerical features. Co-occurrences are not able to distinguish word relations and therefore use vague relations.

Apart from the linguistic categorization one important factor for all features is the required data to learn them. Initially, all methods use one-hot encoded, concatenated vectors, or conceptually similar representation as a

feature. The feature learning transforms those simple features into more sophisticated ones with lower dimensionality to avoid overfitting and to reduce training time. The thesauri-based approaches mostly rely on manual feature engineering. Therefore, it does not use feature learning at all. Subsequently, no additional training data is required. The landscape of distributional features is more divided. Traditional models use unsupervised methods, like PCA or SVD to reduce the dimensionality. Word embedding use language models and thereby are also unsupervised. Finally, typical deep learning approaches, like CNNs or RNNs integrate the feature learning into the classification task. Essentially each layer of those networks represents features. To learn those features labeled data is required. The features are closely related to the classification task they are trained on and do not provide any theoretical insight.

4.1. Data Pre-processing. The pre-processing for extracting the NLU feature has two main objectives: Separating the text into features units and cleaning the input data. The separation is arguably required for extracting any feature based on the introduced notion of hierarchical semantics. This tokenization is independent of the actual features. It provides the elements to calculate those features. On the contrary data cleansing relies on the actual extracted features, because it changes key characteristics of the dataset to increase the generalizability and adapts to the requirements of the feature extraction models.

- Tokenization: Splitting the input text yields the parts used to calculate features. The introduced general pipeline to extract NLU features uses words as atomic units and groups them into sentences and finally into a larger utterance. Thereby tokenization must divide the text into words, sentences, and in some cases into larger utterances like paragraphs. Since the separation into paragraphs is not required for the short texts analyzed during this study it is not described in detail. Additionally, paragraphs depend heavily on the used convention, e.g. two-line feeds as a separator. Those conventions lack ambiguity and are consequently easy to separate.
- Data Cleansing: One crucial part of all practical machine learning tasks is data quality. Typical real-world data has a low quality due to missing measurements, faulty measurements or simply inaccurate measurements. These errors introduce unwanted noise. Therefore, data cleansing is required to improve the quality and boost the outcome. Textual data is no exception to this general rule. In addition to data specific cleaning steps, textual data faces commonly a problem with dimensionality. Two properties of the text contribute to this problem. On one side some words occur frequently in almost any text without adding information on the other side some words are so

unique that they do not occur in other texts. Both aspects lead to a larger vocabulary and consequently to more dimensions in the initial BoW model. Therefore, the curse of dimensionality is supplementary handled by removing and generalizing specific tokens.

- Token Removal: Removing irrelevant or redundant features is frequently used for most ML methods. At this point, tokens are the most basic features of the text. Consequently, irrelevant tokens are removed before more complex features are calculated. The removal of tokens depends on the actual task and is thereby a domain-specific task. The three token types (Numbers, Special Characters, and Stop words) are commonly removed. Regular expressions are used to remove number tokens and tokens containing special characters. Stop words are removed based on an NLTK corpus containing 179 stop words.
- Token Generalization: Comparable to the aggregation of data objects it is possible to reduce the distinguished words by grouping them. Tempus, numbers, and other grammatical details are in most cases not important to determine the overall meaning of a paragraph accordingly those lemmas can be transformed into the corresponding lexeme. The recovering of lexemes is called lemmatization. The word usage and the context are analyzed to determine the lexeme. For example, POS tags have to distinguish between meeting as a verb or a noun because the lexeme of the verb is met but the noun should not be changed at all. This kind of morphological analysis is a complex and time-consuming task, therefore stemming is often considered as a heuristic alternative. Stemming only analyses the word itself and tries to reduce the word to its stem. This stem differs from the lexeme. Lemmatization focuses on converting lemmas to lexemes by removing infections while stemming focuses on grouping the words with a similar meaning and therefore removes also derivational suffixes. Stemming analyses single words with rule-based approaches to determine the stem.

5. Lexical Approaches. The TF-IDF model is used as a baseline due to its frequent usage in topic clustering/classification for many IR tasks. Furthermore, it does not measure any semantic information aside from the information entropy of each occurring word. This conceptual difference to all other models makes TF-IDF suitable to provide conclusions about the advantages or disadvantages of semantic models in general. This baseline model is accompanied by word2Vec [5] and Global Vectors (GloVe) [6] models on the lexical level. Undoubtedly both models involve semantic knowledge within the word representation and use dense vectors. Similar to the TF-IDF approach both

models are popular for many NLP tasks, especially ANN methods apply frequently Word2Vec due to the possibility to use backpropagation [5]. Both approaches are considered because the influence of the global optimization used by GloVe and the local context window of word2Vec is not foreseeable. This word embedding also builds the fundament for the other two presented models, because those advanced models use the same lexical representation and expand it by taking compositional semantics into account.

As mentioned earlier, lexical methods do not use the word order of a text. According to this sentence and paragraph representation consider the word representations respectively sentence representations as an unsorted list [9]. Since no additional dependencies must be considered the main task for both steps is normalization. This maps the sets with different cardinalities to a fix-sized vector. Therefore, the models become invariant to text lengths. TF-IDF uses averaging, a particularly often used way to normalize data. This is frequently used for word embedding as well. Cer et al. [8] provide an example of this on the STS task. Consequently, averaging the word representation to calculate the sentence and paragraph representation is also used in this work. The resulting process pipeline for all lexical approaches is depicted in Figure 2. The sentence and paragraph representations are calculated by averaging the word representation, where N denotes the total number of words in the paragraph and U_i is the i^{th} word representation. And S_i averaging the word representation to calculate the sentence.

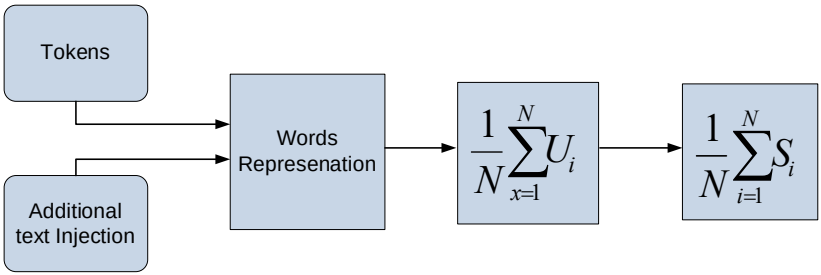


Fig. 2. Process pipeline for lexical approaches

5.1. Word Representation. Naturally, the word representation is the defining element of pure lexical approaches. The three considered models are default approaches. Those basic concepts are not altered, but to be improved, the representations aspects regarding the vocabulary and training are discussed in the following paragraphs.

- Vocabulary: TF-IDF uses a weighted one-hot encoding for words. This requires a fixed vocabulary size otherwise each word would be

represented by a vector with a different length. This vocabulary can be built by analyzing the corpus. Including all words creates, on the one hand, the most accurate representation but on the other hand, the representations get sparse. As mentioned, data sparsity is an undesirable property for further analysis [12]. Based on this trade-off the definition of the vocabulary is a crucial part of the word representation for TF-IDF. Considering the most frequent words will decrease the dimensionality and overall, the least amount of words are ignored. On the other side frequent words do not carry the same information value, therefore stop words are also removed from the vocabulary. The dimensionality of word embeddings is independent of the vocabulary size. Accordingly, no fixed size vocabulary is required. The trade-off between vocabulary size and vector dimensionality does not exist. Consequently, the removal of stop words or less frequent words is unnecessary. This allows adding new words to the vocabulary. The previous weights are simply appended and trained again.

- Training: In opposition to the one-hot representation word embeddings are defined by their occurrence. Therefore, the corpus must contain a sufficient occurrence of each word to define the lexical-semantic of it. A large amount is more likely to fulfill this requirement, but it also contains word ambiguity. Improving the training without introducing more ambiguity has to append the corpus with domain-related data. This work gathers additional data by identifying domain-related Wikipedia articles based on the nouns of the corpus. The extraction utilizes the POS tagger of Natural Language Toolkit (NLTK) to identify those nouns and the Python library to gather articles of those identified nouns.

5.2. Paragraph Representation. The next processing step generates a representation of larger words [8]. Pure lexical approaches normalize the word representations to generate length invariant features. As mentioned, this normalization step calculates for all three approaches the average value defined as in (1):

$$\overrightarrow{w_d} = \frac{1}{n} \sum_{i=0}^{n} \vec{u}_i , \qquad (1)$$

where n denotes the total number of words in the paragraph and $\overrightarrow{u_i}$ is the i^{th} word representation. This calculation creates especially for lexical-semantic encoded in word embedding's two problems.

By averaging contradicting words, the meaning of each word gets annihilated. Since oxymora are used to emphasize certain aspects, this would not be a desirable result. Due to the un-specific notion of word relations used by word embedding's an oxymoron is just an artificial example for two appropriate reverse vectors. Avoiding this behavior requires a more complex model that considers compositional semantics.

Another aspect that is not accurately modeled by simple averaging word representations is the individual influence of words. Depending on the POS, word position, and the occurrence the meaning of each word has to be weighted differently. For example, nouns and verbs contribute more to the overall meaning than adjectives. Partially the IDF factor applied in TF-IDF addresses this problem. Consequently, the averaging of word embedding could be improved by including similar weights as well [5]. This extension is defined in (2).

$$\overrightarrow{w_d} = \frac{1}{n}\sum_{i=0}^{n} IDF(i, D)\overrightarrow{u_i}, \quad (2)$$

where D is the word representations.

5.3. Implementation. Two lexical approaches (*TF-IDF, GloVe*) are implemented from scratch with Python and in the case of *GloVe* with TensorFlow. The Word2Vec representation is calculated by the Gensim Python library. All paragraph representations are calculated with a trivial NumPy averaging function. Therefore, just the word level implementation is discussed briefly.

- GloVe: The Global Vector (GloVe) model is implemented with TensorFlow. Similar to the one-hot encoding a word is mapped to one index which refers to the actual embedding. The TensorFlow graph receives a matrix of indices for context words, center words, and the corresponding co-occurrence counts as input [5]. The center words behave like context words, on that account they are omitted. The corresponding embeddings are generated and randomly initialized with a value between -1 and 1. Based on the index matrix a pointer to the related embedding is generated for each matrix element.

- Par2Vec: One model using the word order is the unsupervised Par2Vec model introduced by Pajak et. al [3]. This method extends the language model of Word2Vec to additionally optimize a paragraph

representation. Instead of just using the word embedding-s to predict words for each paragraph a vector is added to the word embedding-s and is subsequently also altered during the optimization. The Par2Vec models assume with this additional vector for each paragraph that the accurate prediction of a word does not only depend on the corresponding context/center words but also the overall topic of this paragraph. Therefore, the vector distinguishes between paragraphs with different topics. Due to the simultaneous calculation, both representations influence on each other. Therefore, the word embedding of this model differs from the simple lexical approaches. Par2Vec considers the word order based on the context windows, but it does not use any syntactical structure other than unordered neighboring words. According to this property, it is controversial whether Par2Vec can be considered a sequential mode as shown in Figure 3.

Similar to the Word2Vec model the Par2Vec representation is implemented by the Gensim Python library. Following the usage of this library is described, which does not differ greatly between Word2Vec and Par2Vec. Gensim renames the Par2Vec model to Doc2Vec, which established it as a synonym for Par2Vec. The Gensim approach is oriented on the Scikit-learn classifier. One classifier object is created with all hyper-parameters, trained and finally, the classifier predicts the results.

Both models (Word2Vec, Par2Vec) require a special input format of tagged words/paragraphs. This pre-processing step creates trainText as a list of named tuples. For each element, the words and unique tags are required to train the embedding. The invert-vector function does not need this tag later on.

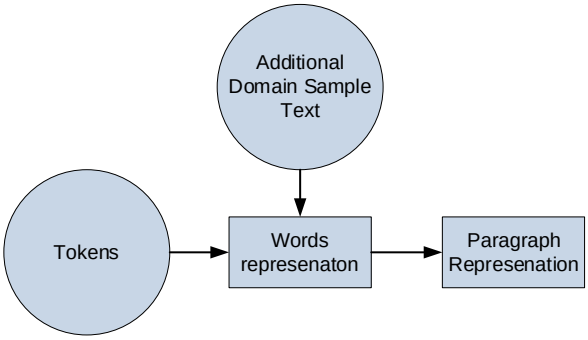


Fig. 3. Process pipeline for the Par2Vec model

6. Transfer-Learning Approach: BiLSTM. The last reviewed feature is the most elaborated model concerning the syntactical structure of a text. Instead of using context windows similar to Par2Vec the presented model includes the full word order and represents thereby a typical sequential model on the sentence level. As such it can enclose the linguistic structure of a sentence, like dependencies to gather compositional semantics. This approach extends the lexical model (word2Vec or GloVe) by replacing the word averaging on sentence-level with an RNN which is trained to provide compositional semantics. Likewise, a brief look at the processing pipeline Figure 4 shows this difference.

The sentence representation or to be more precise the ANN has to be trained to extract compositional semantics. This internal training task must include semantic knowledge. Through the training on such a task, the parameters of the ANN learn to create an accurate sentence representation that captures the sentence's meaning. Afterwards, the trained model can generate a representation of all kinds of sentences. The learned knowledge from a specific task is transferred. In opposition to the Par2Vec and word, embedding approaches this training data is not provided by large unlabeled corpora based on language models. Hence the model is supervised [14].

The two key factors of this transfer-learning approach are the model and the initial task. Especially the initial task must rely on semantic knowledge. One example of such a task is the STS task. It provides due to the SemEval workshop sufficient labeled data and benchmark models. Furthermore, the task of classifying sentences due to overall similarity resembles the later contemplated problem of detecting duplicates. Based on the training task the BiLSTM model proposed in [4] for STS is considered as classifier during this work. This BiLSTM model shows on the STS benchmark a reasonably good performance. Supplementary LSTM approaches are commonly used for many NLP tasks due to the variable input length. However, the focus of this work lays on the principal application of transfer-learning, therefore evaluating a great number of different models is out of scope for this paper.

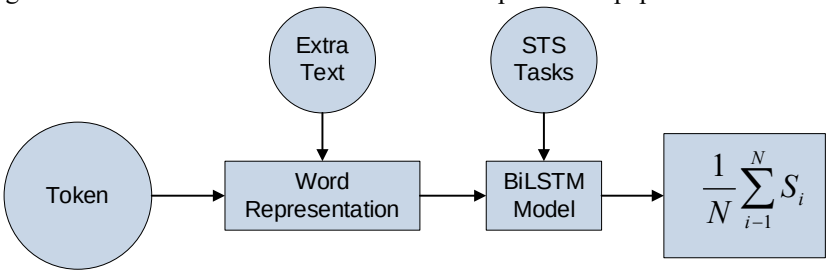


Fig. 4. Process pipeline for the BiLSTM model

6.1. Sentence Representation. The sentence representation is generated by training the weights and biases of a BiLSTM model to predict the semantic similarity between two sentences. The intermediate representation of this training task constitutes an NLU feature for each sentence, which includes compositional semantics and can be applied to tasks with less or none training data. The process of the whole training task as proposed in [4] (Fig. 5). The intermediate representation of the two sentences is compared with a simple feedforward network with one hidden layer. The resulting probability vector is compared to the actual label to calculate the loss and adapt the parameter of the feedforward model and the BiLSTM. Both components are in detail introduced in the following paragraph.

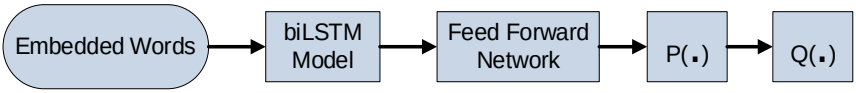


Fig. 5. The network architecture of all components for the STS training task

The input sequence of word embeddings is considered in the forward direction and backward direction to avoid influencing the hidden state excessively by the end or beginning of the sequence. Each direction is fed into one single layer LSTM cell. This constitutes the BiLSTM. The last hidden state of both cells is concatenated and forms the final output of the BiLSTM component.

The feedforward network compares two sentences regarding the similarity of the semantic annotated representations. To avoid training the feedforward network to append the semantic sentence representation, the inputs of this component are two simple difference measurements instead of the actual vectors [16]. The semantic representation should be exclusively generated by the BiLSTM component because the feedforward network is exchanged later on. Consequently, all linguistic knowledge incorporated into it would be lost. In theory, using a different value as input separates the semantic representation and the similarity classification. Due to the difficult interpretation of ANN transformations, this cannot be proven trivially. Therefore, the sentence vectors calculated by the BiLSTM are compared regarding the absolute difference in each dimension and the angular difference. Both measurements are defined as:

$$\overrightarrow{h_+} = |\vec{h}_L - \overrightarrow{h_R}|; \tag{3}$$

$$\overrightarrow{h_+} = \overrightarrow{h_L} \theta \overrightarrow{h_R}, \quad (4)$$

where $\overrightarrow{h_L}$ respectively $\overrightarrow{h_R}$ denotes the sentence representations and θ denotes element wise multiplication.

According to Tai et al. a combination of both comparisons is empirically proven to be superior compared to either of them alone [13]. Thereby, two hidden layer combines $\overrightarrow{h_x}$ and $\overrightarrow{h_t}$ by adding both transformed vectors *W* as formalized in (5).

$$\overrightarrow{h_S} = \sigma[W_{\overrightarrow{h_S}} \overrightarrow{h_x} + W_{\overrightarrow{h_i}} \overrightarrow{h_t} + b_{h_x}] \ . \quad (5)$$

Finally, the output layer calculates the probabilities for each similarity class of the STS task by applying a softmax nonlinearity, therefore the result vector is defined as $pin[0,1]$and the output layer is shown in (6):

$$\hat{p} = soft\,max[W_{\overrightarrow{h_S}} \overrightarrow{h_S} + b_y]. \quad (6)$$

Logically the predicted class is defined as $\hat{y} = [0,1,2,3,...,5] \times \hat{p}$. By this definition, the probabilities are used as weights to sum the classes and average them. This allows the average label a continuous result instead of a strict classification into the six classes.

The actual loss is calculated between those two probabilities as cross-entropy and optimized by simple gradient descent. Due to many parameters of the model, it tends to overfit, therefore, an *L2* regularization term is added to the cost function. This term penalizes large weight values by adding the sum of quadratic weights. Additionally, the BiLSTM applies dropout to introduce noise to avoid overfitting. Merkx et al. [4] demonstrated that no significant improvement for the STS task, but due to the focus on transfer-learning and the reported improvements for the sentimental task dropout regularization is applied in this model.

The BiLSTM approach is implemented like GloVe with TensorFlow. The BiLSTM component is assembled out of standard TensorFlow classes for LSTMs, calculating dropouts and passing a bidirectional sequence to RNN cells. Those abstractions allow it to implement it within a few lines.

The cross-entropy losses are optimized with a standard TensorFlow gradient descent algorithm. The required probability for the actual label is

calculated. The two equal comparisons mask the two nearest classes to the actual label and split the floating-point part between those classes.

The detection of duplicates as a binary classification problem can be evaluated by the Receiver Operating Characteristic (ROC) curve. This metric is commonly applied to evaluate a binary classifier. The basics of the ROC are briefly described in the following section. This default model is complemented with a view into the distribution of similarity between duplicates and non-duplicates.

The binary classification can be divided into four possible situations for each tuple:

- True Positive (TP): The text tuple is labeled as duplicate and classified as such.
- False Positive (FP): It is not labeled as duplicate but classified as one.
- True Negative (TN): The tuple is not labeled nor classified as duplicate.
- False Negative (FN): It is labeled as duplicate but classified as non-duplicate.

7. Evaluation and Results. The following section discusses the evaluation details for each NLU feature. All features utilize here established threshold classifier in combination with cosine similarity. The ROC curves adjust the threshold value in 0:01 steps starting at $T = 0$ *till* $T = 1$. The feedforward network is not used as a classifier due to the small amount of labeled data.

7.1. Pre-Processing. The preprocessing steps stemming and stop word removal aim to reduce the sparsity of the word representations. Therefore, just models that directly utilize BoW word representations can be improved by those methods. Consequently, just the TF-IDF representation is evaluated regarding the preprocessing steps. The result of the Bosch dataset is presented in Figure 6. In the following (Fig. 6), TF-IDF representation does not show any improvement for stemming and the removal of stop words. The reason can be seen by comparing it to a simple TF model. The stemming does not significantly boost the performance of this model either. Most likely the vocabulary size is too small to observe any effect of generalization. Additionally, the reported improvements for other English NLP tasks are quite small [8]. In opposition to the modest results regarding stemming the removal of stop words improves the TF model. Stop words are by definition frequent words which means that the TF-IDF model already reduces the influence of those words with the IDF factor. Hence the expected positive influence of the preprocessing steps to reduce the data sparsity cannot be observed due to the small vocabulary and corpus size. The Wikipedia dataset which is even smaller shows similar behavior. To align

the TF-IDF representation with the other evaluated NLU features both preprocessing steps are skipped in other comparisons.

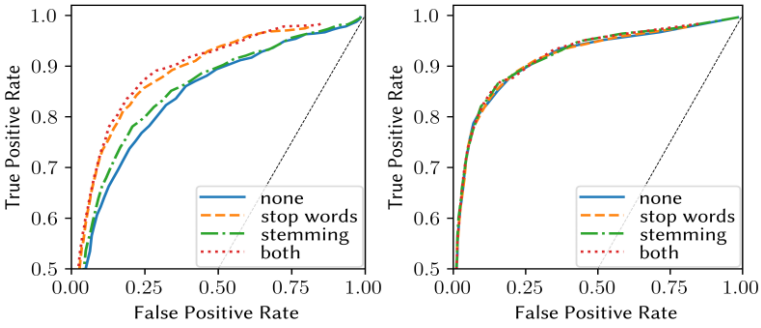


Fig. 6. ROC curves for TF (a) and TF-IDF (b) representations with different preprocessing steps on the Bosch dataset. The y-axis is shifted by 0:5

7.2. Training Data. One important aspect of word embedding is the training corpus. Naturally, embedding's benefit from a large training corpus, since more text provides more information for the word co-occurrences required to define the actual embeddings. On the other side due to word ambiguity and text-specific word usage, the training dataset has to represent the word distribution of the analyzed text as well. Those two requirements often contradict each other. Therefore, a closer look into the influences of training data for detecting duplicates is conducted on the GloVe embeddings. This evaluation requires domain-specific language. Hence it uses the Bosch dataset. Additionally, the retrained GloVe embedding is already trained on the Wikipedia corpus. Therefore, comparing domain-specific training and pre-trained models contain a bias on the dataset.

The used collection of bug reports for the evaluation is provided by Robert Bosch Engineering and Business Solutions Private Limited. These bug reports are collected but due to the nature of those bug reports the limited dataset is made publicly available. It contains precise descriptions of sensitive data about Bosch software. These key characteristics of the used dataset are presented in Table 1.

The pre-trained embedding is trained on a large corpus of Wikipedia articles and the fifth Gigaword corpus which contains text from several news agencies. The overall corpus contains up to 6 billion tokens [11]. The original Bosch dataset contains just 118,835 tokens. Also, a small amount of domain-specific data is extracted from suitable Wikipedia articles. This collection

contains additional 29,318 tokens. The corresponding results of the threshold classification are illustrated in Figure 7.

The dataset contains 13,095 bug reports of which 1,515 reports are marked as a duplicate of at least one other ticket of the dataset. Each report contains a short one-sentence summary, a longer description, and a list of duplicate tickets. For the evaluation, the NLU features are extracted solely from the description. Additionally, there are the duplicate lists used to generate a labeled list with two tickets and a Boolean value to identify those tickets either as duplicate tickets or not. Overall, the 1,515 duplicate reports are transformed into 780 duplicate tuples. The small difference between a tuple and a duplicate number shows that a few reports have more than one duplicate. The remaining tuples are considered as non-duplicates. The tuples are subsampled to improve the ration between duplicates and non-duplicates. This is described as part of the data preparation, after taking a closer look into the properties of the actual descriptions.

Table 1. The most important dataset properties

Property	Bosch bug reports	Wikipedia summaries
Total size	13095 reports	200 summaries
Tuple duplication	780	110
No duplication	3900	550
Data origin	Real	Artificial using methods
Structure	Several paragraphs with Headings, bullet points, etc.	One paragraph with no Structural elements
Language	Keywords, English sentences	Plain English sentences
Vocabulary	Specific domain	No specific domain

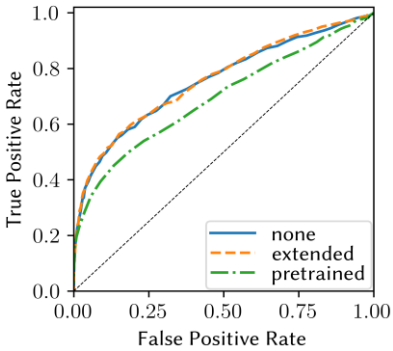


Fig. 7. ROC curves for GloVe embedding trained on different datasets

The Wikipedia dataset contains the summaries of different articles. The summary or lead section of a Wikipedia article is the text before the table of

contents. The summaries are gathered from the English, German, French, and Spanish Wikipedia. All the summaries are translated into English using Google Translate except the English texts. This creates several summaries about the same topic which are consequently labeled as duplicates. To avoid large differences, the text size differs at most 15% between the duplicates and is also manually filtered. The Wikipedia dataset contains just 200 summaries of which each summary is a duplicate. Therefore, it contains 110 duplicate tuples and 550 randomly selected non-duplicate tuples. The dataset is smaller than the bug reports but provides more reliable Duplicate labels and it is more homogenous in language and text structure.

The domain-specific training shows a consistently better result than the pre-trained embedding. Taking a closer look, it indicates two aspects that contribute to this result. Firstly, the domain-specific training captures the actual meaning even with fewer data more accurately. Secondly, from the 11,309 different words in all Bosch tickets, just 8,0085 are also words in the pre-trained dataset. Over 3,000 words remain randomly initialized for the pre-trained embedding. However, the word embedding trained on the extended dataset shows a slight improvement over the plain dataset. The additional domain data just contains around 30,000 tokens. Consequently, the expected improvement is also small. The observed behavior is in line with the expectation and hints that a larger amount of domain data improves the model.

7.3. Comparison of Lexical Methods. Finally, the three introduced word representations are directly compared. The results for both datasets are illustrated (Fig. 8). All features utilized to established threshold classifiers are in combination with cosine similarity. The ROC curves adjust the threshold value in 0.01 steps starting at $T = 0$ till $T = 1$. For the evaluating data, the list of all lexical, Par2Vec and BiLSTM hyper-parameters except for learning parameter is shown in following tables 2-4 given by respectively:

Table 2. List of all hyper-parameters except for the learning parameter

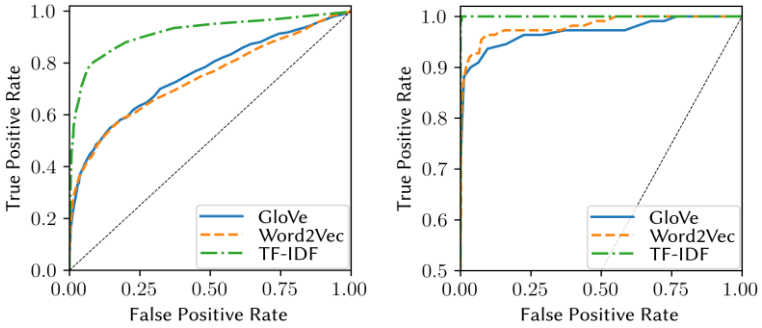
Model	Parametrs	Values
TF-IDF	vocabulary size	9000
Word2Vec	vector size	400
	context window length	5
	subsampling	0.01
	negative samples	5
GloVe	vector size	400
	context window length	5
	co-occurrence maximum	100
	Scaling factor	0.80

Table 3. List of Par2Vec hyper-parameters except for the learning parameter

Model	Parameters	Values
Distributed BoW	vector size	400
	context window length	5
	subsampling	0.01
	negative samples	5
GloVe	vector size	400
	context window length	5
	subsampling	0.001
	negative samples	5

Table 4. List of Par2Vec hyper-parameters with the exception of the learning parameter

Sub-Model	Parameters	Values
Word embedding	GloVe	
BiLSTM	output size	100
	maximum sentence length	60
Feed Forward Network	hidden layers size	40
Regularization	strength λ	10^{-4}
	keep probability dropout	0.05



(a) (b)

Fig. 8. ROC curves for the mean lexical models. The results of the Bosch dataset are illustrated in (a) and the results for the Wikipedia dataset in (b). The y-axis of (b) is shifted by 0:5.

The divergence between embeddings and baseline is almost an order of magnitude greater for the Bosch dataset than for the Wikipedia collection. One key difference between the datasets on the level of lexical semantics is the heterogenic style of the Bosch dataset. The Bosch histogram shows in Figure 9 for non-duplicates a slightly bigger variation.

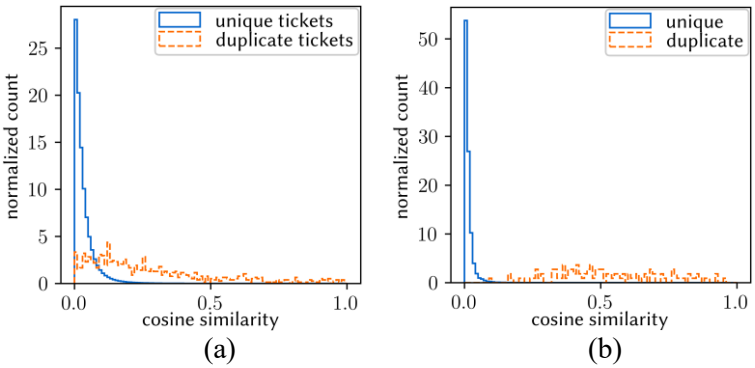


Fig. 9. Histogram of TF-IDF with the (a) Bosch dataset (b) Wikipedia dataset

Instead of full sentences bullet points and keywords occur frequently in the text. In opposition to this, the Wikipedia dataset only consists of complete English sentences. Possibly this lack of structure deteriorates the performance of the embeddings. The TF-IDF features are even able to classify the Wikipedia dataset almost perfectly. For $T = 0.1$ the recall is 0.99 with a precision of 0.98. Naturally, the classification of the Wikipedia dataset is easier, because of the lack of an overall topic between all documents like the software application by bug reports. The low threshold indicates that most of duplicates of the Wikipedia collection are not similar at all. Due to the high variance in both duplicate groups the prediction accuracy decreases significantly based on this difference.

7.4. Par2Vec. The first NLU feature taking compositional semantics into account is Par2Vec. The ROC curve of those unsupervised features is depicted in Figure 10.

Figure 10 shows for both models of Par2Vec that are an improvement compared to the simple averaging of the lexical approaches. They are on a similar level than the baseline TF-IDF although both features are still slightly worse. Interestingly the simpler distributed BoW model for the Bosch dataset is even better than the more complex distributed memory feature. This emphasizes the text structure again. The distributed BoW feature does not use a window to define context words. The representation is optimized to predict all the words of one paragraph. Consequently, the feature is more suitable for the keyword-oriented language of the Bosch dataset. This also indicates word embedding with larger context windows is more robust against ungrammatical sentences.

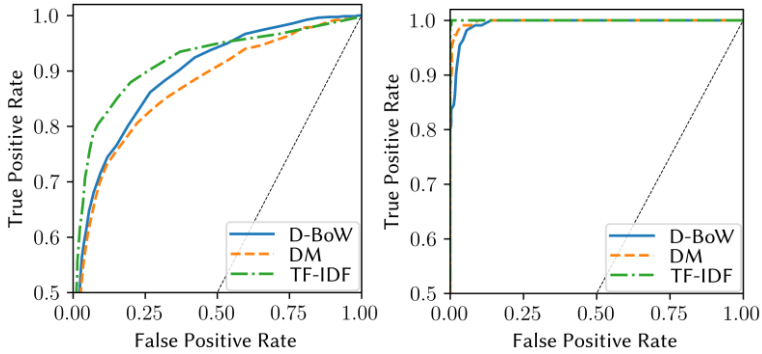


Fig. 10. ROC curves for the distributed BoW (D-BoW) and the Distributed Memory (DM) approach of the Par2Vec model. The results of the Bosch dataset are illustrated in (a) and the results for the Wikipedia dataset in (b). The y-axis is shifted by 0:5

The ROC curve of the Wikipedia dataset does not provide any information about the models due to the principal simple classification task. But the similarity histogram allows based on the variance which is the variance of unique documents from the distributed memory model bigger and the duplicate variance smaller (Fig. 11). This Histogram result confirms that the slightly better ROC curve as well from the previous method.

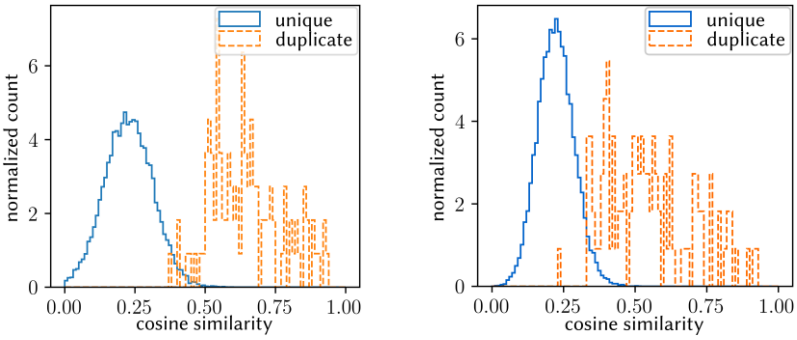


Fig. 11. Histogram of the distributed (a) memory (b) BoW model

7.5. BiLSTM. The last evaluated feature is the transfer-learning approach. The resulting ROC curves are illustrated in Figure 12. The BiLSTM features show overall the worst performance on the Bosch dataset and improve the mean word-embedding model insignificantly on the

Wikipedia dataset. The sentence representations are learned with STS sentences. The text structure of the Bosch dataset does not resemble those sentences. Logically all possible learned linguistic rules cannot be applied. This example shows the obvious limitations of transfer learning in a completely unsupervised setting. However, the performance on the dataset collection, which has a similar sentence structure of the STS training task, shows that the feature learned just to classify the STS sentences. The model can generalize the learned representation to solve other tasks. However, the small difference between both datasets shows that at least a few linguistic rules have been learned. This indication that BiLSTM model with more training data could extract a larger amount of generalized linguistic rules.

After evaluating the ROC curve for the BiLSTM model, the next step is to evaluate BiLSTM features concerning overall performance on the Bosch and Wikipedia datasets and calculate the error rate for the wrong training of data sets. To create the confusion matrix, test highlights datasets information are displayed in Figure 13, that possess the data about analysis amid anticipated and fixated group classes in both datasets. Figure 13 presented the confusion matrix for 3 procedural phases of training and testing separately.

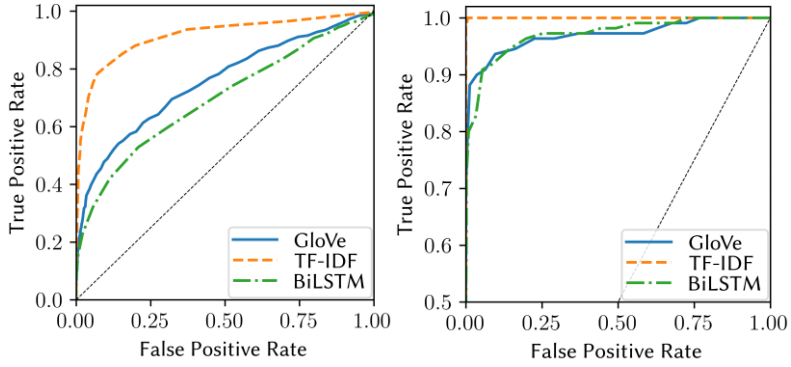


Fig. 12. ROC curves for the BiLSTM model. The results of the Bosch dataset are illustrated in (a) and the results for the Wikipedia dataset in (b). The y-axis of (b) is shifted by 0:5

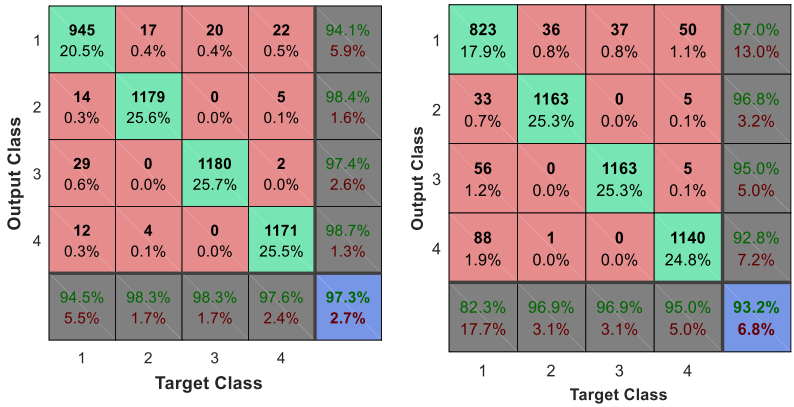


Fig. 13. Performance confusion matrix for the BiLSTM model (a) Bosch datasets (b) Wikipedia datasets

As an anticipated class, it shows all sample feature classification sets. As per Figure 13, the green cells show the successfully classified datasets with patterns and red cells show the unsuccessful classified datasets, and grey cells show the cumulative result of result row and column wise. Finally, blue cell shows the accuracy parentage and error rate that are found during training, test and classification phases.

Referring to Table 1, the blue cell indicates the general ratio of tried cases that were characterized accurately in green and the other way around in red which can learn the structure of a sentence to improve the classification with this knowledge. It also shows that BiLSTM model with more training data could extract a larger amount of generalized linguistic rules with less error rate compared with other methods.

8. Conclusion and Future Work. This paper focused on comparing NLU concepts for featuring semantics of text and applying it to IR. Following this goal, the main contribution of this paper is a comparative study for semantic similarity measurements to identify useful patterns and heuristics for specific IR tasks with varying datasets. This paper compared NLU features for the usage in IR systems. Therefore, several features, which encounter semantic knowledge, were identified, categorized, and described. Those features were evaluated by detecting duplicates in two datasets with different text structures. Due to direct comparison between documents the detection of duplicates is a perfect example for the grouping and order commonly used in IR. Lexical features and Par2Vec were not able to outperform the TF-IDF baseline. Additionally, both models were influenced by the text structure. For well-defined English sentences, both models

performed well. The additional compositional semantic incorporated in Par2Vec also improved the performance. The transfer-learning feature was not able to improve the classification compared to word embedding approaches on documents consisting of English sentences and performed even worse on documents with a mixed text structure. This indicates that the BiLSTM can learn the structure of a sentence to improve the classification with this knowledge. BiLSTM model with more training data could extract a larger amount of generalized linguistic rules. In summary, the results demonstrate the TF-IDF feature, an impressive result on both datasets with reasonable vocabulary size. The evaluation results show that further work is necessary to improve NLU features for the field of IR to achieve better performances than the TF-IDF feature.

The most obvious further work is gathering of more data which is labeled as duplicates. More labeled data allows better classification models like the drafted feedforward network. Additionally, it will be possible to optimize the hyper-parameter of all models. Due to the skew groups, the labeling of randomly picked tuples out of a collection is difficult. Most likely the creation of a new document as a duplicate is easier. However, labeled data is the bottleneck for this concrete detection task and is not going to improve NLU features in general. Furthermore, other methods like Bidirectional Encoder Representations from Transformers (BERT) and Embeddings from Language Model (ELMo), etc. related algorithms can also be utilized to improve the classification with given sentences knowledge.

References

1. Alexopoulou, T., Michel, M., Murakami, A., & Meurers, D. Task Effects on Linguistic Complexity and Accuracy: A Large-Scale Learner Corpus Analysis Employing Natural Language Processing Techniques. *Language Learning*, 2017. 67(S1), pp. 180–208.
2. Keersmaekers, A. Creating a richly annotated corpus of papyrological Greek: The possibilities of natural language processing approaches to a highly inflected historicallanguage. *Digital Scholarship in The Humanities.* 2019.
3. Pajak, B., Fine, A., Kleinschmidt, D., & Jaeger, Learning Additional Languages as Hierarchical Probabilistic Inference: Insights from First Language Processing. *Language Learning*, 2016. 66(4), pp. 900–944.
4. Merkx, D., & Frank, S. Learning semantic sentence representations from a visually grounded language without lexical knowledge. *Natural Language Engineering*, 2019. 25(4), pp. 451–466.
5. Huang, F., Ahuja, A., Downey, D., Yang, Y., Guo, Y., & Yates, A. (2014). Learning Representations for Weakly Supervised Natural Language Processing Tasks. *Computational Linguistics*, *40*(1), pp. 85–120.
6. Kozachok, A. V., Kopylov, S. A., Meshcheryakov, R. V., Evsutin, O. O., & Tuan, L. M. An approach to a robust watermark extraction from images containing text. *SPIIRAS Proceedings*, 2018. *5*(60), 128 p.

7. Nazari, P., Khorram, E., & Tarzanagh, D. Adaptive online distributed optimization in dynamic environments. *Optimization Methods and Software*, 2019. pp. 1–25.
8. Altaf, S., Waseem, M., & Kazmi, L. IDCUP Algorithm to Classifying Arbitrary Shapes and Densities for Center-based Clustering Performance Analysis. *Interdisciplinary Journal of Information, Knowledge, And Management*, 2020. 15, pp. 091–108.
9. Chen, R., Dai, R., & Wang, M. Transcription Factor Bound Regions Prediction: Word2Vec Technique with Convolutional Neural Network. *Journal Of Intelligent Learning Systems and Applications*, 2020. 12(01), pp. 1–13.
10. Mitra, B., & Craswell, N. An Introduction to Neural Information Retrieval t. *Foundations And Trends, In Information Retrieval*, 2018. 13(1), pp. 1-126.
11. Savyanavar, P., & Mehta, B. Multi-Document Summarization Using TF-IDF Algorithm. *International Journal of Engineering and Computer Science*. 2016.
12. Liang, P. Learning executable semantic parsers for natural language understanding. *Communications of the ACM*, 2016. 59(9), pp. 68–76.
13. Berant, J., & Liang, P. Imitation Learning of Agenda-based Semantic Parsers. *Transactions of the Association for Computational Linguistics*, 2015. 3, pp. 545–558.
14. Merkx, D., & Frank, S. Learning semantic sentence representations from a visually grounded language without lexical knowledge. *Natural Language Engineering*, 2019. 25(4), pp. 451–466.
15. Roberts, L. Individual Differences in Second Language Sentence Processing. *Language Learning*, 2012. 62, pp. 172–188.
16. Dontsov, D. O. Algorithm of thesaurus extension generation for enterpise search. *SPIIRAS Proceedings*, 2014. 7(30), 189. p.
17. Aswani Kumar, C., Radvansky, M., & Annapurna, J. Analysis of a Vector Space Model, Latent Semantic Indexing and Formal Concept Analysis for Information Retrieval. *Cybernetics And Information Technologies*, 2012. 12(1), pp. 34–48.
18. Ch, A. Latent Semantic Indexing based Intelligent Information Retrieval System for Digital Libraries. *Journal of Computing and Information Technology*. 2006.
19. Susanto, G., & Purwanto, H. Information Retrieval Menggunakan Latent Semantic Indexing Pada Ebook. *SMATIKA JURNAL*, 2018. 8(02), pp. 74–79.
20. Blynova, N. Latent semantic indexing (LSI) and its impact on copywriting. *Communications And Communicative Technologies*, 2019. (19), pp. 4–12.
21. Rataj, Karolina. "Electrophysiology of Semantic Violations and Lexical Ambiguity Resolution in Bilingual Sentence Processing." Bilingual Lexical Ambiguity Resolution, 2020. pp. 250–72.
22. Qu, C., Yang, L., Qiu, M., Croft, W. B., Zhang, Y., & Iyyer, M. BERT with History Answer Embedding for Conversational Question Answering. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2019.
23. Ghavidel, H., Zouaq, A., & Desmarais, M. Using BERT and XLNET for the Automatic Short Answer Grading Task. *Proceedings of the 12th International Conference on Computer Supported Education*. 2020.
24. Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W., Choi, Y., Zettlemoyer, L. QuAC: Question Answering in Context. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018.
25. Reddy, S., Chen, D., & Manning, C. D. CoQA: A Conversational Question Answering Challenge. Transactions of the Association for Computational Linguistics, 2019. 7, pp. 249–266.
26. Sur, C. RBN: Enhancement in language attribute prediction using global representation of natural language transfer learning technology like Google BERT. *SN Applied Sciences*, 2019. 2(1).

Saud Altaf —Ph. D., Pir Mehr Ali Shah Arid Agriculture University. Research interests: the area of wireless sensor networks, artificial intelligence, HCI, and visible light communication for research, teaching, and learning. The number of publications — 35. SAUD@uaar.edu.pk, www.uaar.edu.pk; Office 21, UIIT, Main Murree Road, Rawalpindi, Pakistan; office phone: +923130009013.

Sofia Iqbal — MPhil., Pakistan Space and Upper Atmosphere Research Commission (SUPARCO), Pakistan. SOFIAIQBAL.SUPARCO@gmail.com; www.suparco.gov.pk; Sector-H, DHA Phase II, Islamabad, Pakistan; office phone: +923213399831.

Muhammad Soomro — Professional Engineer and Member of Engineering New Zealand. The number of publications — 22. MWASEEM@manukau.ac.nz, www. manukau.ac.nz; Gate 1, L Block Main Reception, Newbury Street, Otara MIT Manukau: Ground floor, MIT Manukau, Auckland, New Zealand; office phone: +64221323717.

УДК 006.72 **DOI** 10.15622/ia.2021.3.5

С. АЛЬТАФ, С. ИКБАЛ, М. СООМРО

ЭФФЕКТИВНЫЙ АЛГОРИТМ КЛАССИФИКАЦИИ ЕСТЕСТВЕННОГО ЯЗЫКА ДЛЯ ОБНАРУЖЕНИЯ ПОВТОРЯЮЩИХСЯ НЕКОНТРОЛИРУЕМЫХ ПРИЗНАКОВ

Альтаф С., Икбал С., Соомро М. **Эффективный алгоритм классификации естественного языка для обнаружения повторяющихся неконтролируемых признаков.**

Аннотация. Эта статья фокусируется на том, чтобы уловить смысл значения текстовых функций понимания естественного языка (NLU) для обнаружения дубликатов неконтролируемых признаков. Особенности NLU сравниваются с лексическими подходами для доказательства подходящей методики классификации. Подход трансфертного обучения используется для обучения извлечению признаков в задаче семантического текстового сходства (STS). Все функции оцениваются с помощью двух типов наборов данных, которые принадлежат отчетам об ошибках Bosch и статьям Википедии. Цель данного исследования-структурировать последние исследовательские усилия путем сравнения концепций NLU для описания семантики текста и применения их к IR. Основным вкладом данной работы является сравнительное исследование измерений семантического сходства. Экспериментальные результаты демонстрируют результаты функции Term Frequency–Inverse Document Frequency (TF-IDF) для обоих наборов данных с разумным объемом словаря. Это указывает на то, что двунаправленная долговременная кратковременная память (BiLSTM) может изучать структуру предложения для улучшения классификации.

Ключевые слова: кластеризация, информационный поиск, функция TF-IDF, Par2Vec, тексты на естественном языке, лексические подходы

Сауд Альтаф — Ph. D., Сельскохозяйственный университет Пир Мехр Али Шах Арид. Область научных интересов: беспроводные сенсорные сети, искусственный интеллект, человеко-машинное взаимодействие и связи видимого света. Число научных публикаций – 35. SAUD@uaar.edu.pk, www.uaar.edu.pk; офис 21, Равалпинди, Пакистан; р.т.: +923130009013.

София Икбал — M. Phil, Пакистанская Комиссия по исследованию космоса и верхних слоев атмосферы (SUPARCO), Пакистан. Область научных интересов: беспроводные сенсорные сети, искусственный интеллект, человеко-машинное взаимодействие и связи видимого света. sofiaiqbal.suparco@gmail.com; www.suparco.gov.pk; Сектор-H, фаза II DHA, Исламабад, Пакистан; р.т.: +923213399831.

Мухаммад Соомро — член инженерного общества Новой Зеландии. Область научных интересов: беспроводные сенсорные сети, искусственный интеллект, человеко-машинное взаимодействие и связи видимого света. Число научных публикаций — 22. MWASEEM@manukau.ac.nz, www.manukau.ac.nz; выход 1, главная стойка регистрации блока L, Ньюбери-стрит, Оклендский технологический университет, Окленд, Новая Зеландия; р.т.: +64221323717.

Литература

1. *Alexopoulou, T., Michel, M., Murakami, A., & Meurers, D.* Task Effects on Linguistic Complexity and Accuracy: A Large-Scale Learner Corpus Analysis Employing Natural Language Processing Techniques. Language Learning, 2017. 67(S1), pp. 180–208.
2. *Keersmaekers, A.* Creating a richly annotated corpus of papyrological Greek: The possibilities of natural language processing approaches to a highly inflected historical language. Digital Scholarship In The Humanities. 2019.
3. *Pajak, B., Fine, A., Kleinschmidt, D., & Jaeger, T.* Learning Additional Languages as Hierarchical Probabilistic Inference: Insights from First Language Processing. Language Learning, 2016. 66(4), pp. 900–944.

4. *Merkx, D., & Frank, S.* Learning semantic sentence representations from a visually grounded language without lexical knowledge. Natural Language Engineering, 2019. 25(4), pp. 451–466.
5. *Huang, F., Ahuja, A., Downey, D., Yang, Y., Guo, Y., & Yates, A.* Learning Repre- sentations for Weakly Supervised Natural Language Processing Tasks. Computational Linguistics, 2014. 40(1), pp. 85-120.
6. *Kozachok, A. V., Kopylov, S. A., Meshcheryakov, R. V., Evsutin, O. O., & Tuan, L. M.* An approach to a robust watermark extraction from images containing text. SPIIRAS Proceedings, 2018. 5(60), 28 p.
7. *Nazari, P., Khorram, E., & Tarzanagh, D.* Adaptive online distributed optimization in dynamic environments. Optimization Methods and Software, 2019. pp. 1–25.
8. *Altaf, S., Waseem, M., & Kazmi, L.* IDCUP Algorithm to Classifying Arbitrary Shapes and Densities for Center-based Clustering Performance Analysis. Interdisciplinary Journal of Information, Knowledge, And Management, 2020. 15, pp. 91–108.
9. *Chen, R., Dai, R., & Wang, M.* Transcription Factor Bound Regions Prediction: Word2Vec Technique with Convolutional Neural Network. Journal Of Intelligent Learning Systems and Applications, 2020. 12(01), pp. 1–13.
10. *Mitra, B., & Craswell, N.* An Introduction to Neural Information Retrieval t. Foun- dations And Trends® In Information Retrieval, 2018. 13(1), pp. 1-126.
11. *Savyanavar, P., & Mehta, B.* Multi-Document Summarization Using TF-IDF Algori- thm. International Journal of Engineering and Computer Science. 2016.
12. *Liang, P.* Learning executable semantic parsers for natural language understanding. Communications of the ACM, 2016. 59(9), pp. 68–76.
13. *Berant, J., & Liang, P.* Imitation Learning of Agenda-based Semantic Parsers. Transa- ctions of the Association for Computational Linguistics, 2015. 3, pp. 545–558.
14. *Merkx, D., & Frank, S.* Learning semantic sentence representations from a visually grounded language without lexical knowledge. Natural Language Engineering, 2019. 25(4), pp. 451–466.
15. *Roberts, L.* Individual Differences in Second Language Sentence Processing. Language Learning, 2012. 62, pp. 172–188.
16. *Dontsov, D. O.* Algorithm of thesaurus extension generation for enterpise search. SPIIRAS Proceedings, 2014. 7(30),189 p.
17. *Aswani Kumar, C., Radvansky, M., & Annapurna, J.* Analysis of a Vector Space Model, Latent Semantic Indexing and Formal Concept Analysis for Information Retrieval. Cybernetics And Information Technologies, 2012. 12(1), pp. 34–48.
18. *Ch, A.* Latent Semantic Indexing based Intelligent Information Retrieval System for Digital Libraries. Journal Of Computing and Information Technology. 2006.
19. *Susanto, G., & Purwanto, H.* Information Retrieval Menggunakan Latent Semantic Indexing Pada Ebook. SMATIKA JURNAL, 2018. 8(02), pp. 74–79.
20. *Blynova, N.* Latent semantic indexing (LSI) and its impact on copywriting. Commu- nications And Communicative Technologies, 2019. (19), pp. 4–12.
21. *Rataj, Karolina.* "Electrophysiology of Semantic Violations and Lexical Ambiguity Resolution in Bilingual Sentence Processing." Bilingual Lexical Ambiguity Resolution, 2020. pp. 250–72.
22. *Qu, C., Yang, L., Qiu, M., Croft, W. B., Zhang, Y., & Iyyer, M.* BERT with History Answer Embedding for Conversational Question Answering. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2019.
23. *Ghavidel, H., Zouaq, A., & Desmarais, M.* Using BERT and XLNET for the Automatic Short Answer Grading Task. Proceedings of the 12th International Conference on Computer Supported Education. 2020.

24. *Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W., Choi, Y., Zettlemoyer, L.* QuAC: Question Answering in Context. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018.
25. *Reddy, S., Chen, D., & Manning, C. D.* (2019). CoQA: A Conversational Question Answering Challenge. Transactions of the Association for Computational Linguistics, 2019. 7, pp. 249–266.
26. *Sur, C.* RBN: Enhancement in language attribute prediction using global representation of natural language transfer learning technology like Google BERT. SN Applied Sciences, 2019. 2(1).